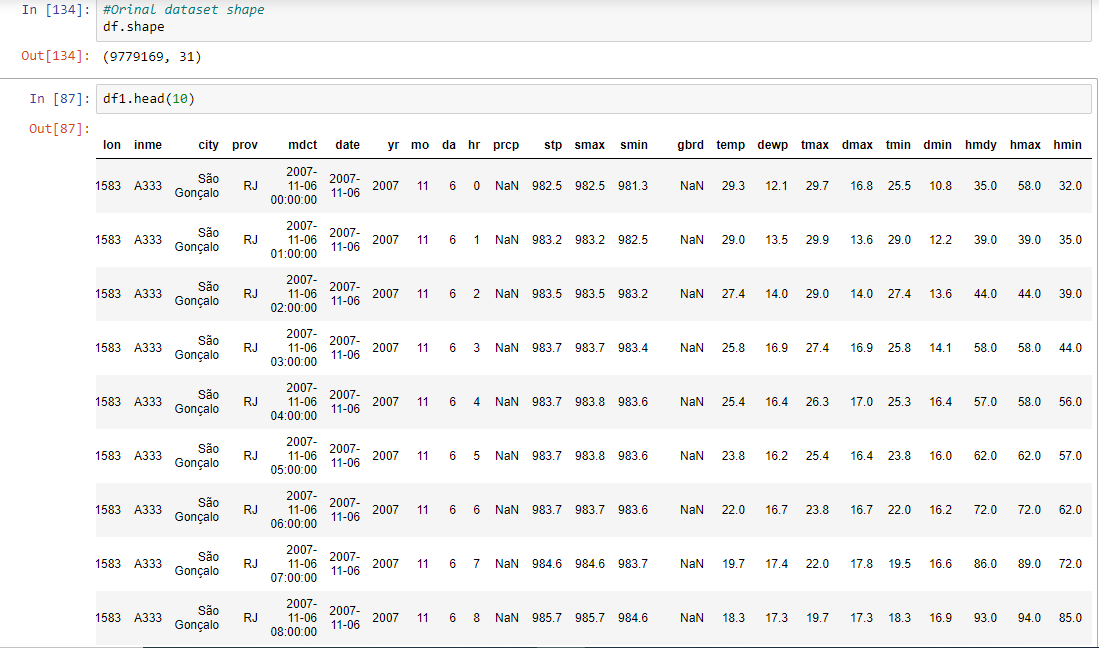
**Observation for Project 2**

**Data Pre-Processing**

We have considered the dataset titled Hourly Weather Surface for Brazia(a Southeast region) published on the largest dataset platform kaggle.com. The overall purpose of this dataset is to analyze, depict and perform pre-processing to convert a noisy dataset to more clear and clean to fit into a data analysis pipeline.

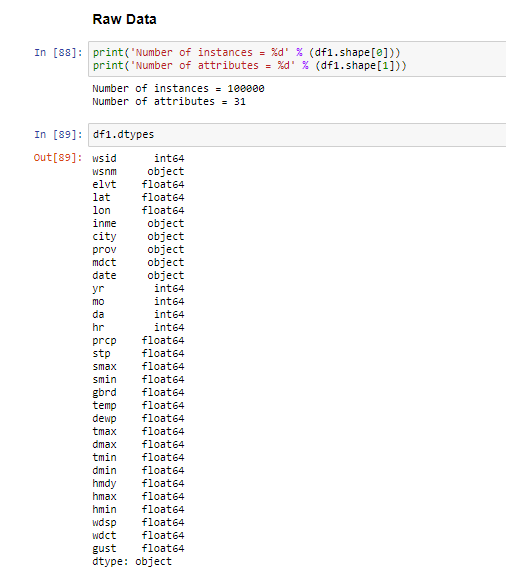
**Dataset**

1. This data set is obtained from Kaggle
   1. <https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region/kernels>
2. The original dataset contains over 9779169 unique rows alongside with 31 attributes indicating the weather station, city, temperature, rain precipitation, dates, humidity etc.
3. Covers hourly weather data from a total of 122 weather stations of southeast region brazil.
   1. Cities like Rio de Janeiro, Minas Gerais e Espírito Santo etc,



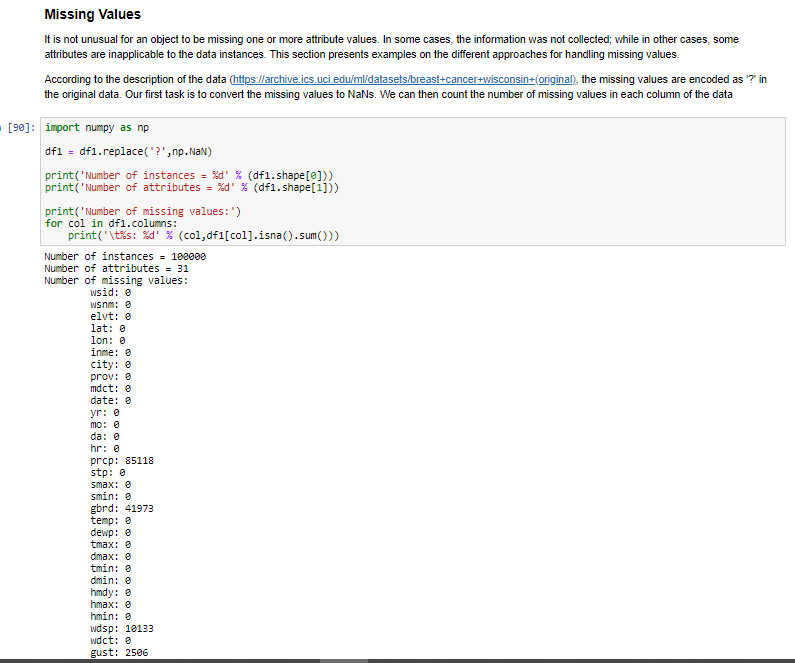
**Raw Data**

* Before applying any techniques for pre-processing the initial steps is visualizing the sort of data set we are given. The initial values for each columns and rows.
* Selecting the first 10000 rows from the original dataset
  + Number of instances that occur within dataset = 10000
  + Number of attributes = 31
* Identifying the data types given. In our observation we find float64 otherwise a float value are dominant within dataset. Followed by object otherwise string values and finally a dew Integer values.
* Floating variables indicate our dataset contains correct weather measurements for each attribute since the dataset is a collection of hourly weather patterns ranging from temp to humidity.



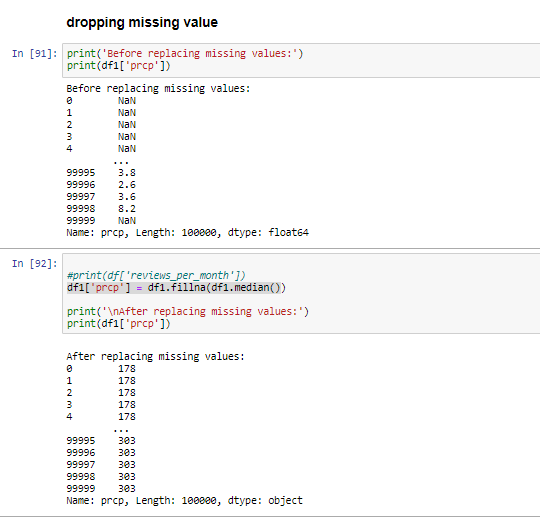
**Identifying Missing Values**

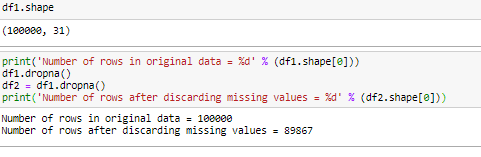
* In order to make a dataset userful and analyzable to apply any models like classification, neural networks etc. the goal is to have none to least amount of missing values in a dataset.
* Missing values can cause disruptions and make dataset noisy to analyze. In our dataset we use a for loop to run through each column values and print the findings to all column total count of missing values.
* From our findings we can observe the following.
  + Prcp - precipitation, gbrd - Solar radiation, wdsp - Wind speed in metres per second, gust - wind gust in metres per second. Contain the most amount of missing value occurrences.



**Dropping Missing Values**

* Upon our finding of missing values comes dropping those NAN values. They can act against us in our further analyzation of our dataset. Therefore elimination those values containing ‘0’ OR ‘nan’ are best to remove.
* For our purpose we will be dropping those values by replacing with our own generated values. By injecting those row values with a median from the overall column.
* *df1['prcp'] = df1.fillna(df1.median())* this code will fill any missing values with the overall median(). To observe the change we generated a before and after output to run through the column and identify if there are any remaining missing values.
* The following is replacing missing values for the column prcp and gbrd columns.

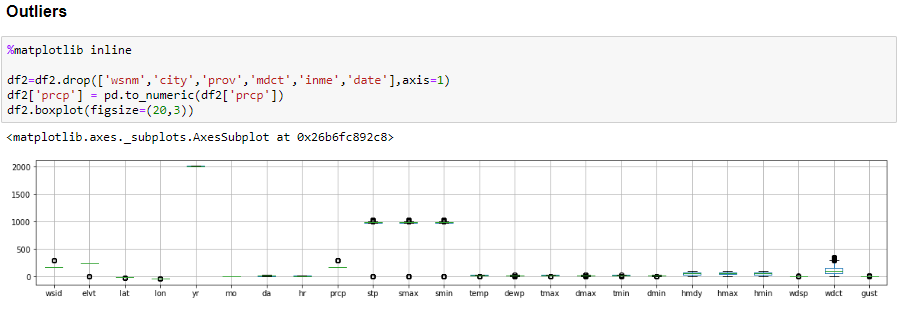
****

****

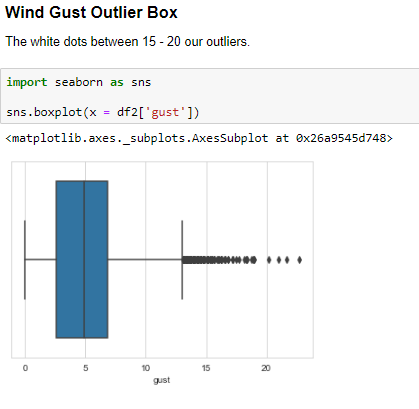
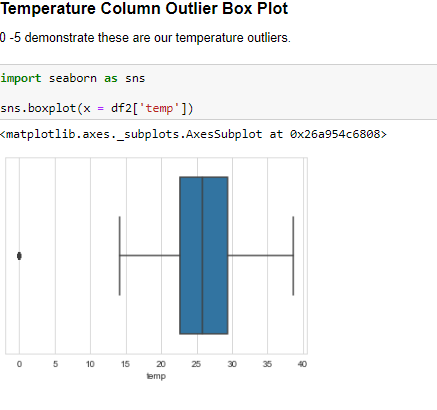
* The above code performs a final check to identify how replacing and dropping those missing values affected the shape of our dataset. Initially we began with 10000 records and after applying the dropa and replace we are left with 89867 records.

**Outliers**

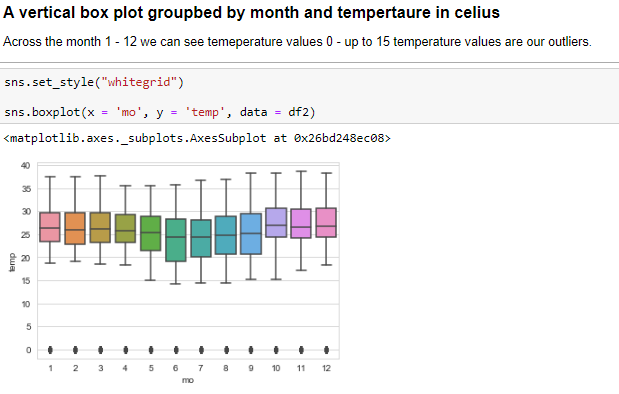
* Having missing values indicates that a dataset can have outliers. Outliers act as data instances containing different characteristics from the rest of the dataset. They tend to be significantly different from the overall points of the dataset.
* To visualize how strong our outliers for our dataset is conducting a boxplot that will run through each column and indicate our outliers.



* The following graphs give a better visualization of the outliers within two columns we choose. These being Temp and Gust. values 0 - 5 for temp values are outliers found in the temperature column. As for wind guest the black dots between 15-20 gust values our outliers found.

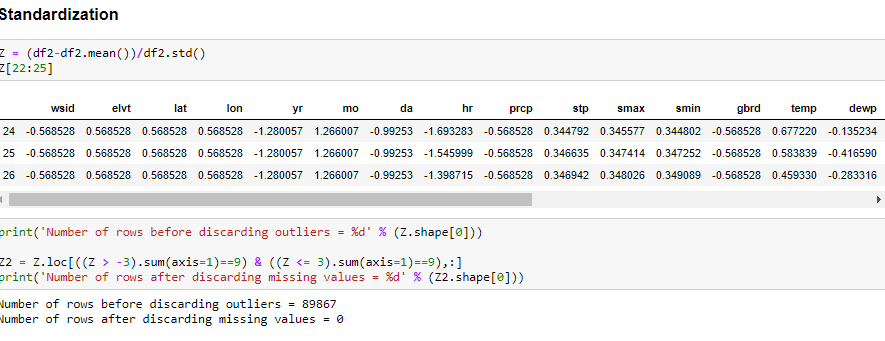


* The following is a box plot for temperature values across a 12 month period.



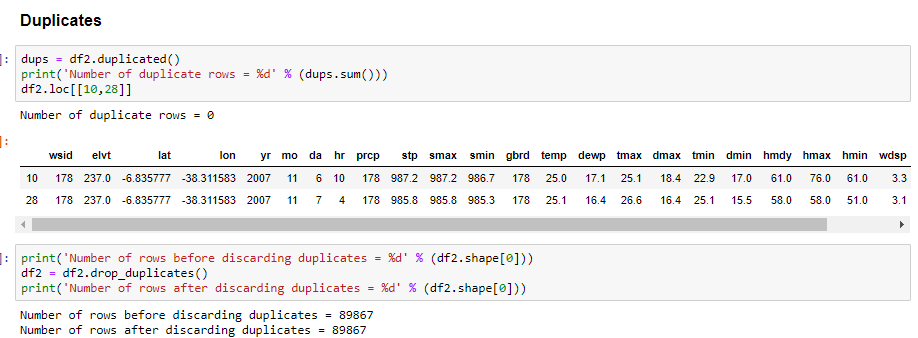
**Standardization**

* Standardizing our dataset allows us to compare features otherwise attributes between for example two columns that contain different units. This allows us to apply them to models for a better analyzation of our dataset and accurate findings.
* For our dataset we are standardized across all data attributes. This will also allow us to remove any outliers based upon our visualization findings of box plots.
* Our findings report before applying standardization we see 89867 outliers and after standardizing we have 0 remaining.

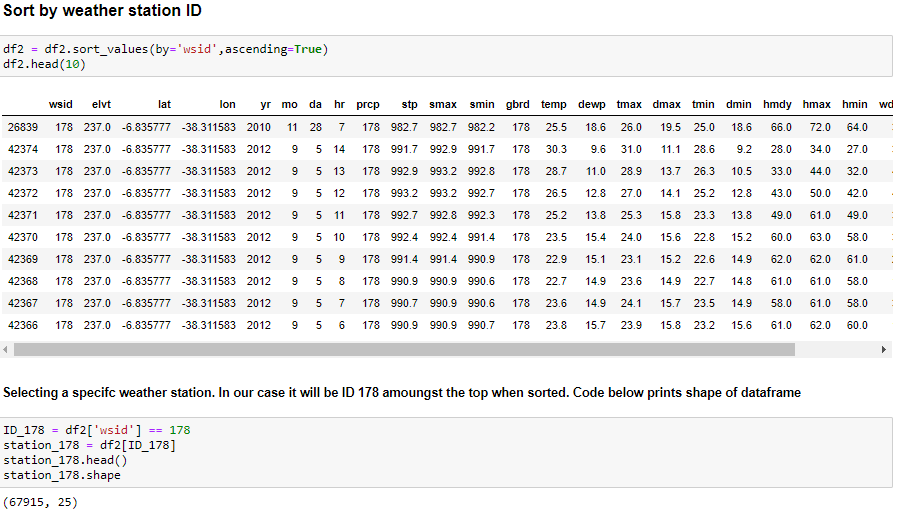


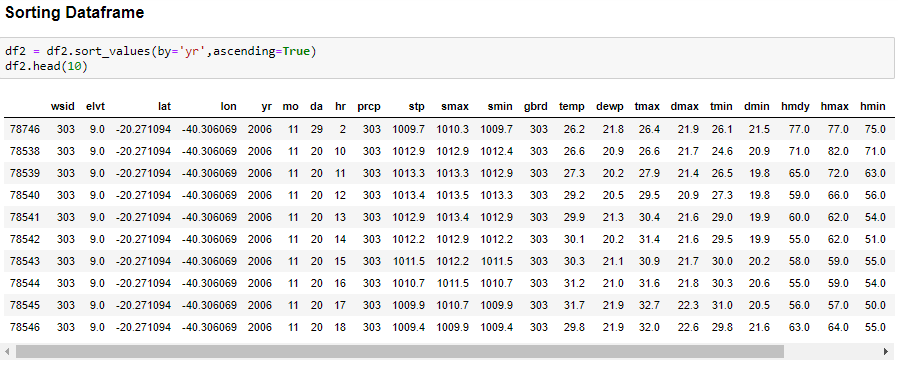
**Identifying Duplicates**

* Another technique to apply for obtaining a clean dataset is removing any duplicate values. Having multiple occurrences can also make our data very noisy for accurate findings.
* Using the following function .duplicated() allow for identification of multiple occurrence within a dataframe
* Our findings indicate we may have removed duplicated in the process of dropping, replacing our standardizing resulting in no duplicated for our output.



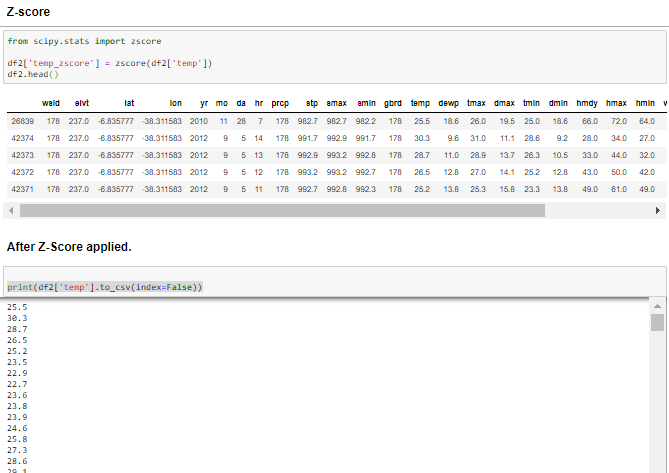
**Sorting DataFrame**

* Sorting our dataset allows for us to visualize our top values. This tool is more for analyzation of our dataset.
* Sorting by Weather Id we can identify the most occuring weather station based on there Column ID
* We can select the top most occuring weather station ID being 178 ****
* **Sorting by Year we can identify the most occuring year for the southeast region**

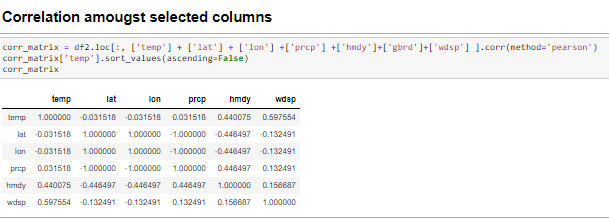
****

**Feature Normalization Z-score and Correlation**

* Also called feature scaling replaces values with their corresponding z-scores.
* To calculate the Z-Score you need to also calculate the mean and standard deviation.
* We applied the z-score function to the following columns temp - temperature and prcp - precipitation



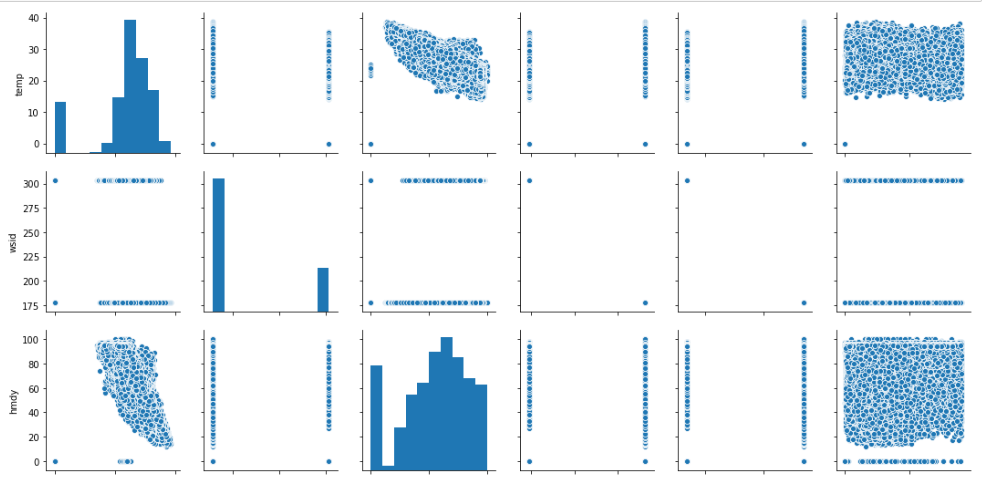
* Using the corr() otherwise known as correlation allows to find the correlation or relationship among the columns selected from the dataset also called using the pearson method.



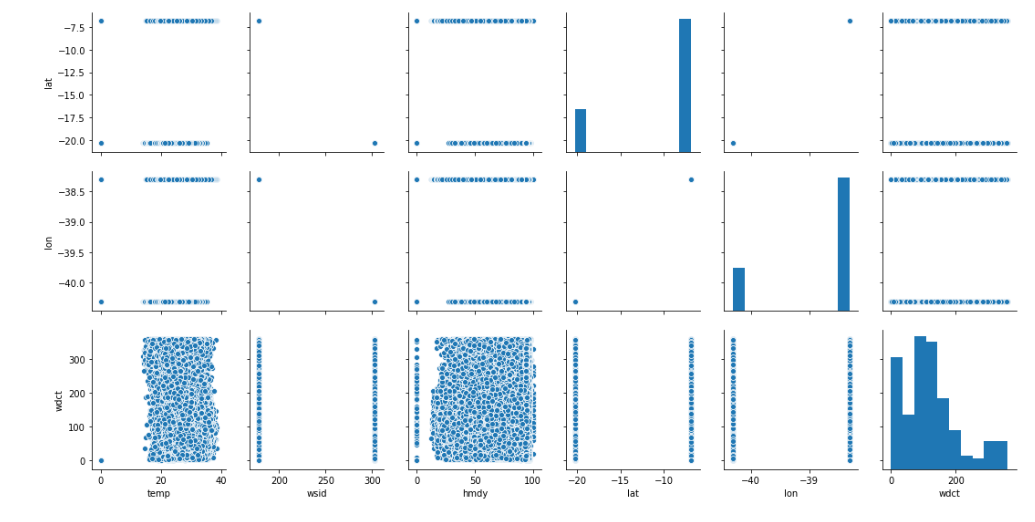
* Using the corr\_matrix data frame in which we grouped the correlation values of the selected columns we can visualize what those correlations depict using the function pairplot()



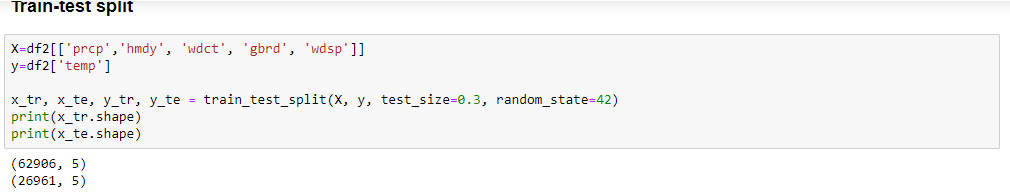
* **Our visual Correlation findings:** 
  + Once we have applied some cleaning against our dataset, the next step is Data Analysis. Identifying what our data is depicting such as finding any relationships, patterns, etc.
  + The first pair plot contains temp, wind and humidity columns



* The next three our pair plots for latitude, longitude, wdct

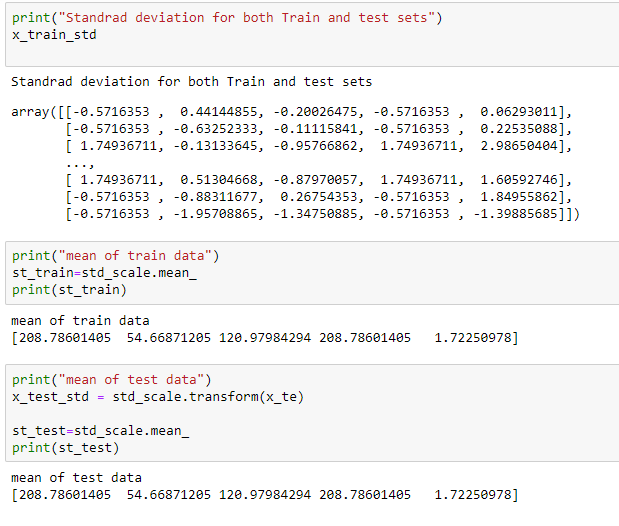


**Train-Test Split**

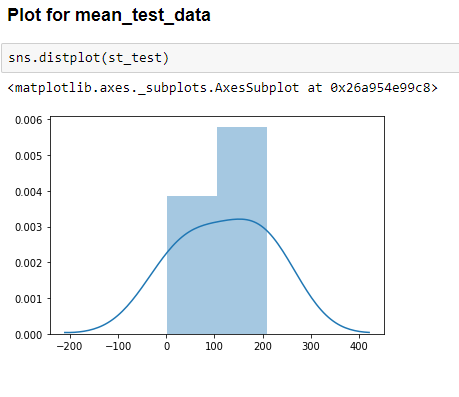
* Training - Test Split is a machine learning model where our predominant findings come from training our data and evaluating the tested otherwise our validated data to compare too.
* To conduct this we have to split according to the ratio for training and validating. Choosing that split is based upon your dataset and what best fits your form.
* Our Training dataset included the columns prcp, hmdy, wdct, gbrd, wdsp
* The main reason for the 30% partitioning of train-testing data is to get best calibration.Usually training-testing data has either 80-20, or 75-25.Our dataset is big and we are taking 70-30 so that we can test the data more accurately, by giving 30% data for testing.
* 

**Mean and Standard Deviation for Dependent Variables**

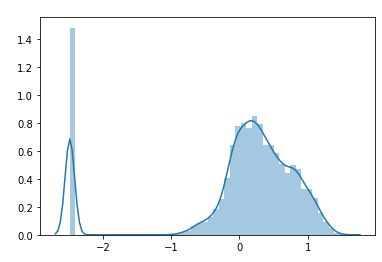
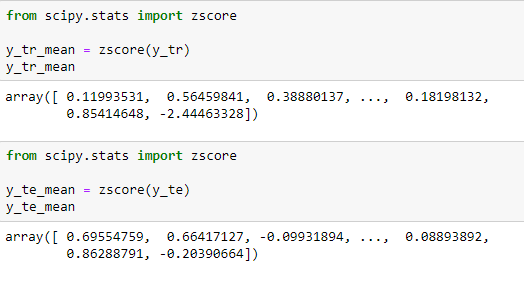
* Using our Train-Test data set we were able to apply simple mean and standard deviation across those newly trained and tested data. This would allow us to compare how distinct those values have changed after applying the training and testing. The following are the results :



* To visualize our mean for our Test data we created a plot graph for those results below :

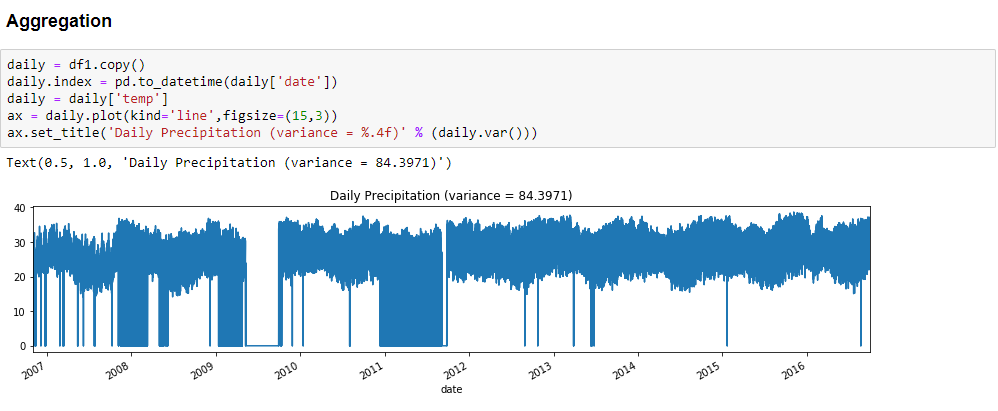
****

* **Mean for our Target Variable:** y = ‘temp’ By applying the z score function across our y train value we are able to see the values for the mean. In addition to the y test mean values result. Next to those mean values is a visual depiction in plot form for the means discovered.
* The standard deviation is the same for variables that are normalized and different for variables that are unnormalized.
* The deviation for test dataset is not much high compared to train dataset

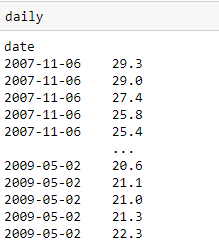
****

**Aggregation**

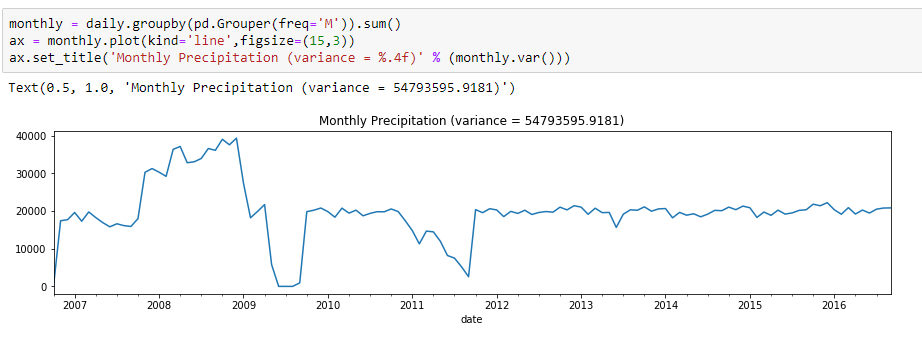
* Data aggregation is another preprocessing task we applied. Our selection of attributes combined in to one single object. This helps reduce the size and improve clarity and stability amongst our dataframe and a fine analysis
* The code we applied was a plot graph of the daily precipitation for a given date since the originality of the data frame is an hourly record of the southeast region Brazil weather findings. One of those being it precipitation values across several cities.



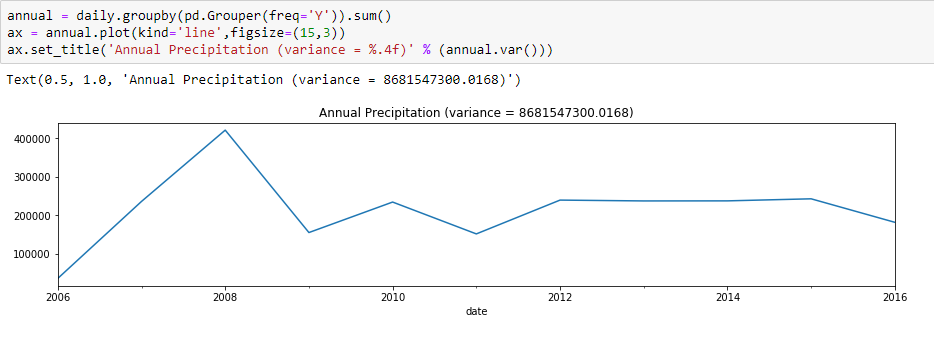
* For a better observation of those prcp values across selected dates recorded see below:



* Below we apply some additional aggregation grouped by month to obtain the total monthly precipitation values

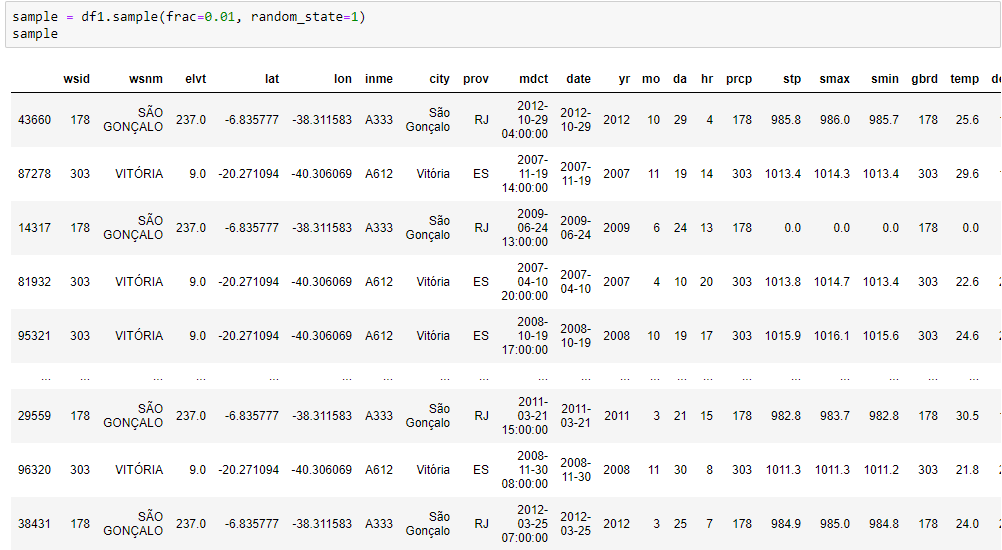


* Below we apply aggregation using the column yr otherwise indicated as year and obtain the annual precipitation values.



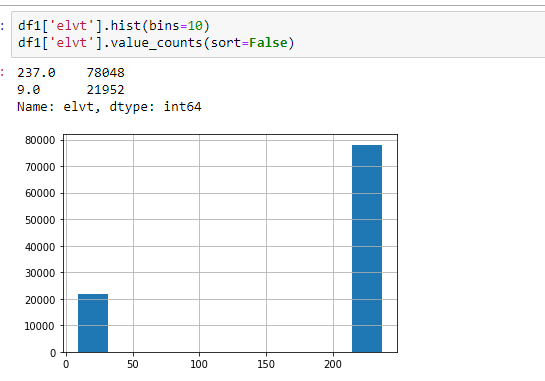
**Sampling**

* Sampling is another pre-processing approach that is used to reduce data for data analysis at a smaller shape such as applying algorithms. Visualizing smaller dataset of the larger dataset given. Allowing for easier processing when data load time is effecting certain approaches. The approach below will be sampling with replacement and without replacement.
* Below we sample first a set of *.sample(3)* 3 random row values.
* The next is selecting *only 1%* of our date without replacement and displaying those selected values.

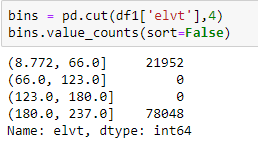


**Discretization**

* Discretization is a data preprocessing step that is often used to transform a continuous-valued attribute to a categorical attribute.
* For this visualization we apply the count frequency for the column elv - elevation



* For equal width, we apply the .*cut()* method to discretize the attribute into 4 bins of the same interval width. Each bin contains the .*value\_counts()* otherwise the number of occurrences for that given column.



**Possible Effects on predictions - Train and Test Dataset**

* This dataset helps in predicting the future weather.
* The dataset has many independent variables [features] which while trained using any model will have accuracy measured.
* For a prediction to have high accuracy there should be minimum error on the target variable [dependent feature] so this is the reason for preprocessing the dataset.

**Why Split the dataset?**

* For this dataset to predict temperature based on the humidity level, solar radiations, wind flow and few other variables**.**
* To find the accuracy of how correct the dataset has predicted we need to split the dataset.

**Conclusion**

Our dataset in the beginning was a challenge to understand and visualize being as it records were large and contained a lot of weather pattern records that to some of us was new to learn. After understanding the sort of data we were given we were able to follow a step by step process to pre-processing data. We learned about the different approaches to dwell deep into identifying what records worked well and those that did not work well for our final analyzation of the overall data. From removing values, working with duplication, standardization, replacing values to tune our date as well as working with visual plot graphs to help visualize outliers and which values made our data more or less clean. Pre-processing is in its own when working with a noisy dataset and visualization in our case helped to understand what sort of records were unnecessary and could be removed to make our analysis the best.